一种云存储环境下的资源调度改进算法 *

徐建鹏，李　欣，赵晓凡

(中国人民公安大学 信息技术与网络安全学院，北京 102628)

摘　要：如何将用户的海量数据以最小的耗时存储到数据中心，是提高云存储效益、解决其发展瓶颈所需考虑的关键问题。首先证明了云存储环境下资源调度方案的存储最小耗时问题属于一个 NPC 问题，再针对现有算法对存储调度因素考虑不全面、调度结果易陷入局部最优等问题，提出了一种全新的资源调度算法。该算法首先利用三角模糊数层次分析法全面分析调度影响因素，得到存储节点的判断矩阵，用于构造后续的遗传算法目标函数，再将简单遗传算法从解的编码、交叉变异操作及致死染色体自我改善等角度进行创新，使其适用于云存储环境下的大规模资源调度。最后与 OpenStack 中的 Cinder 块存储算法及现有改进算法进行了分析比对，实验结果验证了所提算法的有效性，实现了更加高效的资源调度。

关键词：云存储；资源调度；遗传算法；三角模糊数；层次分析法

中图分类号：TP301.6　　doi: 10.3969/j.issn.1001-3695.2018.01.0030

Improved resource scheduling algorithm in cloud storage environment

Xu Jianpeng, Li Xin, Zhao Xiaofan

(*School of Information Technology & Cyber Security*, *People's Public Security University of China*, *Beijing* 102628, *China*)

Abstract: How to store the user's massive data into the data center with the minimum time-consuming is the key issue to be considered in improving cloud storage efficiency and solving the bottleneck of its development. This paper first proved that the minimum storage time-consuming of resource scheduling scheme in cloud storage environment belongs to NPC problem. In view of the incomplete consideration of the existing scheduling algorithms and the problem that the scheduling result tends to fall into the local optimum, a new resource scheduling algorithm was proposed. The algorithm firstly used the triangular fuzzy analytic hierarchy process method to comprehensively analyze the scheduling effecting factors, the judgment matrix of storage nodes was obtained, which was used to construct the follow-up objective function of genetic algorithm, and then the simple genetic algorithm was innovated from the perspective of encoding, cross-mutation operation and self-improvement of lethal chromosome so that it is suitable for cloud storage environment. Finally, this paper analyzed and compared the Cinder block storage algorithm in OpenStack and the existing improved algorithms. The experimental results verified the effectiveness of the proposed algorithm and achieved more efficient resource scheduling.

Key words: cloud storage; resource scheduling; genetic algorithm; triangular fuzzy number; analytic hierarchy process

0　引言

作为云架构的核心支撑，云存储的运行效率直接影响着整个云平台的性能[1]。在现有的主流云存储架构中，如 Apache 的 Hadoop distributed file system[2]、Google 的 Google file system[3]、Amazon 的 Dynamo[4]以及基于 OpenStack[5]开源框架的 Swift、Cinder 和 Ceph，用户的数据通常以较为简单的算法存储到数据中心的存储节点中，如根据剩余存储空间、数据的距离等来进行选择，以此来实现负载的平衡，但并未对网络延迟、带宽、节点内存使用率等其他重要指标进行考虑，这将导致数据存储方案只能陷入一个局部最优解，即并非是最快速的存储方案，而在云存储 PB 级的数据量下，较小的效率提升也会带来可观的收益，提升用户体验。同时，云存储环境下的资源调度也是一种复杂的离散组合最优化问题，即属于带约束的大规模装箱问题，在高复杂度的解空间中，本文无法找到一个多项式时间算法来求其最优解，因此往往采用启发式算法来求近似解。

虽然目前已经有很多解装箱问题的启发式算法，但将其进行创新且应用于云存储环境下资源调度的研究仍然不足。王勇[6]等提出了一种基于可用空间和 I/O 吞吐的多维离线调度算法，降低了活动节点数，提高了资源利用率，但其考虑的指标数不

收稿日期：2018-01-18；**修回日期：**2018-03-05　**基金项目：**中国人民公安大学基础科研经费项目（2016JKF01316）

作者简介：徐建鹏（1993-），男，江西上饶人，硕士研究生，主要研究方向为云计算（cppsujianpeng@126.com）；李欣（1977-），男，副教授，博士，主要研究方向为云计算、网络安全；赵晓凡（1981-），女，讲师，博士，主要研究方向为云计算、网络信息服务.

chinaXiv:201804.02162v1

足，无法实现更全面的分析。同样，肖博[7]在其文章中提出了一种基于三角模糊数的调度算法，实现了较高效的资源调度，但其仅考虑了反指标，且将待存储的数据作为单独的个体来进行决策，未从全局的角度去解该问题，效率较低。

针对上述问题，本文首先在理论上证明了云存储环境下资源调度方案的存储最小耗时问题属于 NPC 问题;其次，使用三角模糊数层次分析法对资源调度所涉及到的各项影响指标进行全面分析，构造出存储节点的判断矩阵，用以表示影响因素之间的重要性关系，该矩阵将成为下一步遗传算法中目标函数的构造依据;再次，云存储规模一般较大，若仍然对解采用一维的二进制编码，则空间效率较低[8]，本文直接利用矩阵作为解的染色体表示，通过矩阵的变换来实现交叉、变异及适应度计算等操作，提高了算法效率且符合实际应用，除此之外，为加速算法的收敛，本文提出了一种基于学习的致死染色体自我改善算法，该算法通过致死个体向优质个体学习的方式来提高整个种群的质量，以实现更快的寻优，同时保证种群基因的多样性。最后，通过与 OpenStack 中 Cinder 块存储算法及文献[7]中算法的实验比较，验证了本文算法的有效性与高效性。

chinaXiv:201804.02162v1

1 相关理论

1.1 NPC 问题

NP（non-deterministic polynomial）是指一类可在多项式时间内，通过数值运算即可解决的问题，即 NP 问题的解能够在一个多项式时间内被很容易地验证，所谓验证，即将一个解代入原问题，可以判断出其正确性。更进一步，如果对于一个 NP 类问题 A，其他所有的 NP 类问题都能够在多项式时间内规约到 A，那么问题 A 就是 NP-complete（NPC）问题。由此，要证明某一个问题 A 是 NPC 问题通常需要两步:a)需证明问题 A 是 NP 问题，即问题 A 的解很容易被验证其正确性；b)构造一个多项式时间内的变换，使得从某一个已被验证过的 NPC 类问题 B 可以很容易地规约到问题 A，那么称问题 A 为 NPC 问题[9]。

1.2 三角模糊数层次分析法

通过把模糊数学[10]的方法引入到层次分析判断矩阵构造中，可以在定量分析时充分考虑到比较判断的模糊性，使判断矩阵能够更好地反映客观情况,同时避免了一致性检验这一步骤，这种方法被称为三角模糊数层次分析法，其定义如下：

定义 1 若对于论域 A 上的模糊集合 $P \in F(A)$，满足条件：存在 $x \in A$ 使得 $\mu_P(x)=1$；同时对于任意的 $y \in (0,1)$，区间 $[x: \ \mu_A(x) \ge y]$ 是闭区间。则有三角模糊数 P，其隶属度函数为

$$\mu_A(x)=\begin{cases}\frac{1}{b-a}\cdot x-\frac{1}{b-a}, x\in[a,b]\\ \frac{1}{b-c}\cdot x-\frac{1}{b-c}, x\in[b,c]\\ 0, \quad x\notin[a,c]\end{cases} \qquad (1)$$

其中: $a \le b \le c$，P 的上界为 c，中值为 b，下界为 a。一般用 (a,b,c) 的形式来表示三角模糊数[7]。

1.3 遗传算法

遗传算法[11]由初始化、个体评价、种群进化三部分构成，其中初始化主要构造种群个体的染色体，即将问题的解进行二进制编码、矩阵编码等；个体评价通过将染色体所代表的解代入目标函数，所得到的结果即是判断种群中个体能否能将基因传递到下一代的重要依据；种群进化包括选择、交叉与变异，是确保基因多样性，跳出局部最优解，找到全局最优解的关键。其原理如图 1 所示。

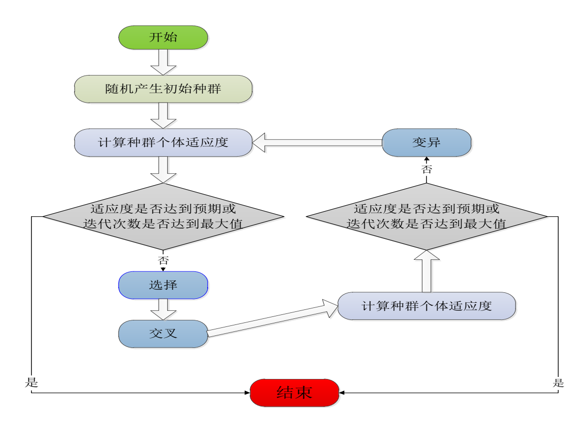


图 1 遗传算法原理图

2 问题的数学模型与分析

2.1 问题建模

云存储环境下的资源调度方案的存储最小耗时问题主要指对同一时刻到达数据中心的待存储数据，采用合适的策略将其存储到节点中，以期得到最小的存储耗时。

本文设待存储的数据序列为 $X=\{x_1,x_2,\cdots,x_m\}$，且 $x_i=\{s_i,m_i,n_i\}$ 分别表示某个数据所需要占用的存储空间、内存大小以及网络资源，设现有的存储节点序列为 $Y=\{y_1,y_2,\cdots,y_n\}$，且 $y_i=\{S_i,M_i,N_i\}$ 分别表示该存储节点所具有的存储空间、内存资源及网络资源。已知某一个存储方案 $K=(k_1,k_2,\cdots,k_m)$，其中 m 表示待存储的数据序号，k_m 的值表示该数据所选择存储节点的序号，对于该方案 K，

chinaXiv:201804.02162v1

需满足如下约束：

$$\begin{cases} S_i \geq s_a + s_b + \cdots + s_c \\ M_i \geq m_a + m_b + \cdots + m_c \\ N_i \geq n_a + n_b + \cdots + n_c \end{cases} \quad (2)$$

即数据占用的资源不能超过存储节点所具有的资源。为充分提高存储效率，本文所考虑的存储耗时指的是每个数据存储的耗时和。

$$T = \min \sum_{i=1}^{m} \left(t_i(\mathrm{s}) + t_i(\mathrm{m}) + t_i(\mathrm{n})\right) \quad (3)$$

其中：$t_i(s)$，$t_i(m)$，$t_i(n)$分别表示该数据存储过程中的数据耗时（由数据的大小、距离所产生的耗时）、内存耗时（由节点的内存情况所造成的耗时）及网络耗时（由节点的网络情况所造成的耗时）。

2.2 调度影响因素分析

根据实际调研情况，本文假设所有的存储节点具有相同的I/O 频率，据此，所需考虑的因素主要包括节点指标、数据指标以及网络指标三个方面，其中节点指标细分为节点故障率、剩余存储空间、内存占用率，数据指标由数据传输距离、数据大小构成，网络指标主要为节点网络带宽与网络延迟，具体如图2 所示。

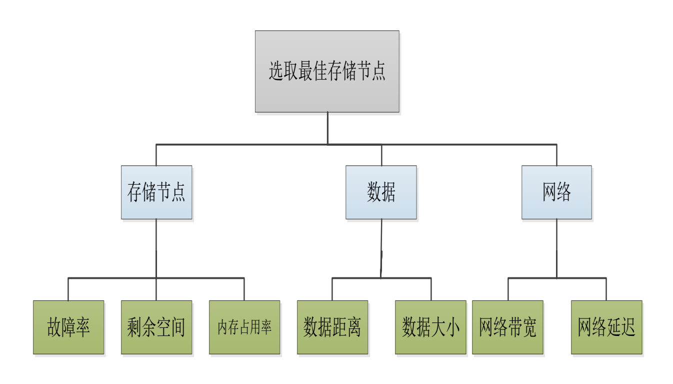


图 2 云存储环境下的资源调度问题所涉及到的指标

本文中，需要调度的资源主要为节点的剩余空间、内存与网络带宽，内存占用率与网络延迟则会随着同时或连续存入的数据个数增加而提高，达到一定程度后会导致数据传输的失败。而其他的调度影响因素，则作为该问题的约束条件，以罚函数的形式体现在目标函数中。

2.3 问题复杂度分析

现已知三维装箱问题属于 NPC 问题[9]，由此给出如下定理：

定理 1 云存储环境下资源调度方案的存储最小耗时问题为 NPC 问题。

证明 a)首先证明上述问题属于 NP 问题。假设随机生成一个存储方案 $K' \in K$，$K' = (k_1', k_2', \cdots, k_m')$，本文只需要验证公式（2）中的各约束，即可在多项式算法时间内来判断该方案是否属于一个可行解，因此该问题属于 NP 问题。

b)将上述问题归约为一个三维装箱问题，待存储的数据 X 可以归约为待放置的盒子，云存储节点 Y 可归约为箱子，存储数据所需的存储空间、CPU 资源、网络资源可归约为盒子的长、宽、高，存储节点的这三类资源可看做箱子的长、宽、高，则耗时归约为已装入箱子内的盒子的总容积。即原问题可转换为：在现有的数据及存储节点下，是否存在一种存储方案使得最终耗时（占用容积）小于 t，即

$$Time(X, Y, K', L) \leq t$$

其中 L 表示相关的约束，由此可以在多项式时间内将三维装箱问题归约到云存储环境下资源调度方案的存储最小耗时问题上，即定理 1 得到了证明。

3 算法实现

3.1 三角模糊数层次分析

本文采取“1-9”标度来衡量指标的重要性，且通过广泛的调研与分析，对每两个调度影响因素的比较都选取了三组三角模糊数据[7]，求其平均可得表 1。

表 1 云存储环境下资源调度的影响因素

调度影响因素	故障率	剩余空间	内存使用率	数据距离	数据大小	网络带宽	网络延迟
故障率	（1, 1,	（1.46, 2,	（0.52, 1,	(1.38, 2,	(0.32, 0.5,	(0.24, 0.33,	(0.38, 0.25,
剩余空间	(0.38, 0.5,	（1, 1,	(0.27, 0.33,	(0.33, 0.5,	(0.62, 1,	(0.28, 0.5,	(0.25, 0.33,
内存使用率	(0.7, 1,	(1.03, 3.03,	（1, 1,	(1.43, 2,	(0.77, 1,	(0.13, 0.5,	(0.32, 0.5,
数据距离	(0.42, 0.5,	(1.03, 2,	(0.40, 0.5,	（1, 1,	(0.78, 1,	(0.14, 0.5,	(0.14, 0.33,
数据大小	(0.96, 2,	(0.75, 1,	(0.76, 1,	(0.76, 1,	（1, 1,	(1.47, 2,	(0.32, 0.5,
网络带宽	(0.93, 3.03,	(1.06, 2,	(1.14, 2,	(1.09, 2,	(0.43, 0.5,	（1, 1,	(0.28, 0.5,
网络延迟	(1.15, 4,	(1.41, 3.03,	(1.39, 2,	(1.2, 3.03,	(1.37, 2,	(1.19, 2,	（1, 1,

chinaXiv:201804.02162v1

根据式（4），求取每个影响因素相对于其他因素的综合重要程度：

$$S_i=\sum_{j=1}^{n}r_{ij}\otimes\left[\sum_{i=1}^{n}\sum_{j=1}^{n}r_{ij}\right]^{-1},i,j=1,2,\cdots,n \quad (4)$$

其中: r_{ij} 表示第 i 个影响因素相对于第 j 个影响因素的重要程度。由此，本文根据表 1 数据进行计算可得:

$$S_1=(5.30,7.08,10.41)\otimes\left(\frac{1}{98.67},\frac{1}{62.69},\frac{1}{37.84}\right)=(0.054,0.113,0.275)$$

$$S_2=(3.13,4.16,6.61)\otimes\left(\frac{1}{98.67},\frac{1}{62.69},\frac{1}{37.84}\right)=(0.032,0.066,0.175)$$

$$S_3=(5.38,9.03,12.06)\otimes\left(\frac{1}{98.67},\frac{1}{62.69},\frac{1}{37.84}\right)=(0.055,0.144,0.319)$$

$$S_4=(3.91,5.83,8.52)\otimes\left(\frac{1}{98.67},\frac{1}{62.69},\frac{1}{37.84}\right)=(0.040,0.092,0.225)$$

$$S_5=(6.02,8.50,11.38)\otimes\left(\frac{1}{98.67},\frac{1}{62.69},\frac{1}{37.84}\right)=(0.061,0.136,0.301)$$

$$S_6=(5.93,11.03,25.09)\otimes\left(\frac{1}{98.67},\frac{1}{62.69},\frac{1}{37.84}\right)=(0.060,0.176,0.663)$$

$$S_7=(8.71,17.06,24.60)\otimes\left(\frac{1}{98.67},\frac{1}{62.69},\frac{1}{37.84}\right)=(0.088,0.272,0.650)$$

定理 2 假设 $P_n=(a_n,b_n,c_n)$，$P_m=(a_m,b_m,c_m)$为两个三角模糊数，$V(P_n>P_m)$表示 P_n 大于 P_m 的可能性[7]，则有

$$V(P_n>P_m)=\begin{cases}\dfrac{a_m-c_n}{(b_n-c_n)-(b_m-a_m)}, & b_n\le b_m,c_n\ge a_m\\ 1 & \text{其他}\end{cases} \quad (5)$$

根据式（5）计算出每个影响因素优于其他因素的纯测量度，可得：$V(S_1\ge S_3)=0.876$，$V(S_1\ge S_5)=0.903$，$V(S_1\ge S_6)=0.773$，$V(S_1\ge S_7)=0.540$，$V(S_2\ge S_1)=0.720$，$V(S_2\ge S_3)=0.606$，$V(S_2\ge S_4)=0.839$，$V(S_2\ge S_5)=0.620$，$V(S_2\ge S_6)=0.511$，$V(S_2\ge S_7)=0.297$，$V(S_3\ge S_6)=0.890$，$V(S_3\ge S_7)=0.643$，$V(S_4\ge S_1)=0.891$，$V(S_4\ge S_5)=0.788$，$V(S_4\ge S_6)=0.663$，$V(S_4\ge S_7)=0.432$，$V(S_5\ge S_3)=0.969$，$V(S_5\ge S_6)=0.858$，$V(S_5\ge S_7)=0.610$，$V(S_6\ge S_7)=0.857$，其余比较值均为 1。

$$d(C_1)=V(S_1\ge S_2,S_3,S_4,S_5,S_6,S_7)=0.540$$

$$d(C_2)=V(S_2\ge S_1,S_3,S_4,S_5,S_6,S_7)=0.297$$

$$d(C_3)=V(S_3\ge S_1,S_2,S_4,S_5,S_6,S_7)=0.643$$

$$d(C_5)=V(S_5\ge S_1,S_2,S_3,S_4,S_6,S_7)=0.610$$

$$d(C_4)=V(S_4\ge S_1,S_2,S_3,S_5,S_6,S_7)=0.432$$

$$d(C_6)=V(S_6\ge S_1,S_2,S_3,S_4,S_5,S_7)=0.857$$

$$d(C_7)=V(S_7\ge S_1,S_2,S_3,S_4,S_5,S_6)=1$$

对其做归一化处理：

$$sum=0.540+0.297+0.643+0.432+0.610+0.857+1=4.379$$

$$d^{'}(C_1)=\frac{0.540}{sum}=0.123$$

$$d^{'}(C_2)=\frac{0.297}{sum}=0.068,\quad d^{'}(C_3)=\frac{0.643}{sum}=0.147$$

$$d^{'}(C_4)=\frac{0.432}{sum}=0.099,\quad d^{'}(C_5)=\frac{0.610}{sum}=0.139$$

$$d^{'}(C_6)=\frac{0.857}{sum}=0.196,\quad d^{'}(C_7)=\frac{1}{sum}=0.228$$

对各个权值做检验有：

$$d^{'}(C_1)+d^{'}(C_2)+d^{'}(C_3)+d^{'}(C_4)+d^{'}(C_5)+d^{'}(C_6)+d^{'}(C_7)$$
$$=0.123+0.068+0.147+0.099+0.139+0.196+0.228=1$$

由此即求得七个指标的相对权重分布，可用于遗传算法目标函数的构造。

3.2 遗传算法的改进

为将遗传算法充分应用于云存储环境下资源调度中，本文提出如下定义：

定义 3 待存储的数据序列为 $X=\{x_1,x_2,\ldots,x_n\}$，可提供存储服务的节点序列为 $Y=\{y_1,y_2,\ldots,y_m\}$，若 x_i 存储在 y_j 中，则 $x_iy_j=1$，否则 $x_iy_j=0$。设矩阵 M_k 表示 X 与 Y 之间的一种存储方案，其中 $M_k^{ij}=x_iy_j$，则可得：

$$M_k=\begin{bmatrix}x_1y_1 & x_1y_2 & \cdots & x_1y_m\\ x_2y_1 & x_2y_2 & \cdots & x_2y_m\\ \cdots & \cdots & \cdots & \cdots\\ x_ny_1 & x_ny_2 & \cdots & x_ny_m\end{bmatrix}$$

且满足：

$$\sum_{j=1}^{m}x_iy_j=1 \quad (6)$$

以上定义表示，在遗传算法的初始化阶段，本文即将种群中所有的个体编码为矩阵 M_k 的形式，与之相对应的，在个体评估阶段，本文首先需要构建云存储环境下资源调度方案的存储最小耗时问题的目标函数 Z_k，可得

chinaXiv:201804.02162v1

$$\begin{cases} Z_k = M_k \otimes P^T + \sum_{i=1}^{n}\sum_{j=1}^{m} x_i y_j \cdot D_j \cdot w_4 + Size \cdot w_5 \\ P = [P_1, P_2, \cdots, P_m] \\ P_j = Space_j \cdot w_2 + Mem_j \cdot w_3 + NetB_j \cdot w_6 + NetD_j \cdot w_7 + Fail_i \cdot w_1 \end{cases} \quad (7)$$

$$\sum_{j=1}^{m}\sum_{i=1}^{n} x_i y_j \cdot Size_i \le Capacity_j \quad (8)$$

其中矩阵 P 表示每个节点的代价矩阵，运算“ $\otimes$ ”表示在基本的矩阵相乘运算的基础上增加更新操作，即当多个数据存储到一个节点中时，需要对节点的各项指标进行更新，得到 $P_j^{'}$，且令 $P_j = P_j^{'}$。D_j 表示第 j 个节点距离数据中心根节点的路由跳数，$Size_i$ 表每个数据的大小，$Capacity_j$ 表示第 j 个节点的初始存储空间，$Space_j$、Mem_j、$NetB_j$、$NetD_j$、$Fail_j$ 分别表示剩余存储空间、内存使用率、网络带宽、网络延迟与故障率的归一化后的得分，本文假设所有的节点具有相同的 I/O 频率，所以从整体的角度来看，数据的大小不会影响调度，因此设置为一个常数 Size， w_i 为各影响因素的权重。为实现种群的进化，需要对种群中的染色体进行交换与变异，根据矩阵 M_k 及式(7)(8)的相关内容，提出如下定义：

定义 4 设存在矩阵 T 为 $n \times m$ 阶初等矩阵，则称种群中个体交换后的结果为 $M_k^{'} = M_k \cdot T$；将染色体 M_k 中若干行中的“1”调整为“0”，并在该行随机选择一个“0”调整为“1”，称之为变异。

通过定义 4 即可在保证式(6)成立的基础上，完成种群的进化。而对于进化过程中所产生的致死染色体，本文提取了一种基于学习的致死染色体自我改善算法，首先有如下定义：

定义 5 种群中适应度排名前 10%的个体染色体为优质染色体 M_N^G；种群中适应度排名后 10%及不符合式（8）的染色体称为致死染色体 M_M^B。

该算法首先为每个致死染色体从优质染色体集合中选择一个作为学习模板，若两者在相同基因位置的值都为 1，则将其保留，然后根据学习率随机从学习模板中选择若干值为 1 的基因，将其替换到致死染色体的对应位置中，则该行原为 1 的基因替换为 0，最后，将所有优质染色体矩阵进行累加，并将累加值等于优质染色体个数的基因进行保留且置为 1，其余置为 0，得到该种群的一个综合优质染色体 M^{AG}，并将 M^{AG} 中值为 1 的基因位置所对应的致死染色体基因位置的值置为 1，相应地该行原为 1 的基因替换为 0。该算法在保证种群基因多样性的基础上提高了基因质量。最终算法在种群的最优染色体的适应度变化小于 1 或者迭代次数大于 100 时停止。其伪代码如下所示：

输入：M_N^G、M_M^B

输出：经过学习后的染色体 $M_M^{B'}$

```
Learnning(M_N^G, M_M^B)
{
M = SUM(M_1^G, M_2^G, ..., M_N^G);
if : M(x_l y_k) == N
M^AG(x_l y_k) = 1;

for(i = 1; i <= M; i++)
{

j = random(N);
if : M_j^G(x_a y_b) == M_i^B(x_a y_b) == 1
M_i^B'(x_a y_b) = 1;
M_i^B'(x_a y_b) = Copy(M_j^G(x_a y_b), StudyRate);        }
if : M^AG(x_c y_d) == 1:
M_i^B'(x_c y_d) = 1;

}
```

4 实验与分析

4.1 实验设计

本文对提出的云存储环境下的资源调度算法的有效性进行分析，与 Cinder 块存储算法及文献[7]的相关算法进行了效率对比，同时实现了代码设计，完成了实验测试。

实验硬件环境：Intel Core i7-6700HQ 四核处理器，8GB 内存，Linux Ubuntu14.04 操作系统。

软件环境：VMware Workstation Pro，C++语言。

实验数据集：本文在程序设计中分别使用 StorageNode 类与 UserData 类的实例来模拟所有的存储节点与待存储数据。StorageNode 类中包含了剩余空间、内存占用率、网络延迟、网络带宽、故障率五个指标，UserData 类中包含了数据大小与距离两个指标项，且在初始化阶段，该七个指标的值均由随机函数产生。

ChinaXiv合作期刊

第一组数据集：实例化生成 1000 个 UserData 类实例及 100 个 StorageNode 类实例，即要在 100 个存储节点中存入 1000 个用户数据。且保证数据总大小小于节点总空间。

第二组数据集：实例化生成 100 个 UserData 类实例及 1000 个 StorageNode 类实例，即要在 1000 个存储节点中存入 100 个用户数据。且保证数据总大小小于节点总空间。

本实验使用两组数据集分别进行 10 组实验，且为确保可比性，每组实验初始状态相同（采用相同的数据实例与节点实例），每组实验中包含了 10 次测试，统计每组实验的平均耗时作为该组对应算法的实验结果，该耗时为算法本身的运算耗时与存储过程产生的耗时之和，其中存储耗时主要由七个影响指标产生。同时，为充分模拟云环境的并行存储模式，所有指标都会随着程序的进行而及时更新，且该程序采用并行方式运行。

4.2　实验结果分析

第一组数据集的实验结果如图 3 所示，其中横坐标表示实验组数，纵坐标表示算法本身的平均运算耗时（s）。

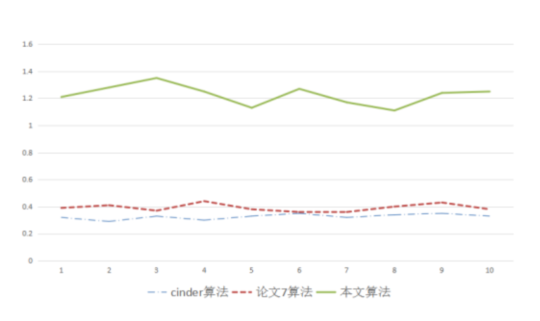


图 3　第一组数据集下三种算法的平均计算耗时比较

相比于 Cinder 与文献[7]算法，本文算法考虑了更加全面的调度影响因素，且使用了改进的遗传算法来求其近似最优解，因此在调度策略的计算上消耗了更多时间。接下来使用第二组数据集来测试算法的整体耗时，即算法运算耗时与执行存储耗时之和，结果如图 4 所示。

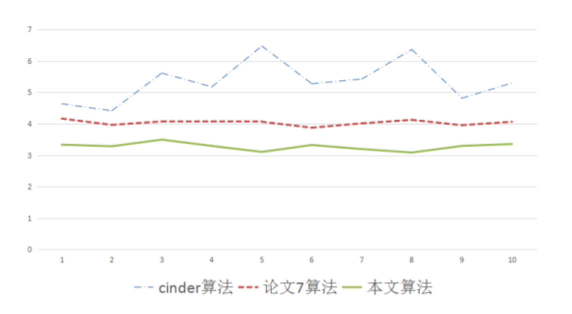


图 4　第二组数据集下三种算法的整体平均耗时比较

由于未考虑其他调度影响指标，数据存入高网络延迟、高内存占用节点等情况会导致耗时的大幅增加，因此 Cinder 算法的表现并不稳定，而文献[7]算法在综合所有正指标后实现了耗时的降低与稳定。本文算法相较另外两种算法，耗时分别降低了 38.7%和 18.8%，实现了更加高效的云存储环境下的资源调度。根据式（3）可知总耗时主要由算法运行耗时及存储过程中的数据耗时、内存耗时和网络耗时组成，本文对第二组实验结果进行深入分析，研究三种算法在存储过程中的耗时分布情况，分析本算法效率提升的直接原因，具体如图 5 所示

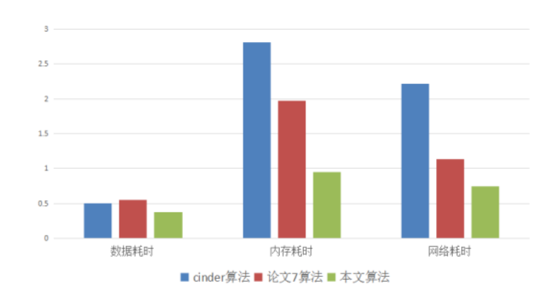


图 5　第二组数据集下三种算法的存储过程中的耗时分布情况

本文算法相较于另外两种算法，效率的提升主要体现在内存耗时与网络耗时上。对于 Cinder 算法，剩余存储空间大的节点可能不具备良好的网络状态或内存状态，从而导致耗时的增加，文献[7]算法由于考虑的指标因素不全面，调度算法未考虑全局最优解，因此也不能很好地提高效率。本文算法在全面分析调度影响因素的基础上采用全局寻优的算法进行求解，充分考虑了各影响因素之间的平衡关系，从而能够最大限度地优化调度方案，实现效率的提升。

为充分分析本文算法，实验将所有的节点状态进行优化，即确保节点内存占用率、网络延迟等指标在初始化时均处于良好状态，使用第一数据集进行十组实验，比较三种算法的总耗时情况，可得图 6。

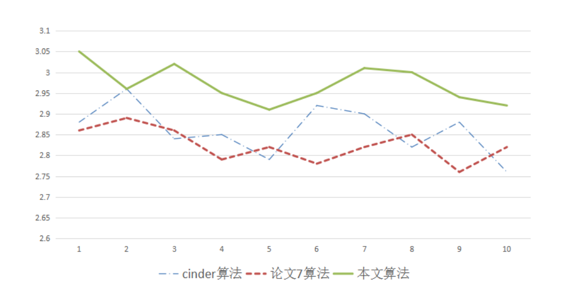


图 6　在节点指标优化的情况下的各算法耗时

由图 6 可得，在存储节点均处于较好的状态时，Cinder 存储将不会存在高内存占用率或高网络延迟的情况，而相较于本文算法的复杂度，Cinder 算法耗时更少，从而在整体上降低总耗时。但云储存规模庞大且复杂，存储节点随时处于读写、删除、数据迁移等操作中，高内存占用与高网络延迟的情况较为普遍，因此本文算法仍具有很好的适用性。

5　结束语

本文针对目前云存储环境下资源调度研究不够深入、调度算法不够完善的问题，提出了一种高效且适用于云存储的全新

chinaXiv:201804.02162v1

ChinaXiv合作期刊

资源调度算法，并对算法进行了实现。实验数据表明本文算法在现有算法基础上实现了效率的提升，具备有效性与高效性，为提高用户体验与云存储效益提供了有力保障。但另一方面，本文算法在调度方案的计算耗时方面仍有待提高，需进一步优化算法流程，从遗传算法的初始种群生成及个体的基因编码结构入手，设计更加快速的种群进化方式，以实现更加稳定、高效的云存储环境下的资源调度。

参考文献：

[1] 徐建鹏，李欣，孙海春. 一种基于可信策略的云存储持久性检测方法 [J]. 计算机应用研究, 2018, 35 (7): 1-2 (2017-07-27) .

[2] Borthakur D. The Hadoop distributed file system: architecture and design [J]. Hadoop Project Website, 2007, 11 (11): 1-10.

[3] Howard S G, Gobioff H, Leung S. The Google file system [J]. ACM SIGOPS Operating Systems Review, 2003, 37 (5): 29-43.

[4] Decandia G, Hastorun D, Jampani M, *et al.* Dynamo: amazon's highly available key-value store [J]. ACM SIGOPS Operating Systems Review, 2007, 41 (6): 205-220.

[5] Jackson K, Bunch C, Sigler E. OpenStack cloud computing cookbook [M]. [S. l.] : Packt Publishing, 2015.

[6] 王勇，田博，王端. 云存储的多维离线调度算法 [J]. 计算机应用与软件, 2017, 34 (6): 309-313.

[7] 肖博. 云存储调度关键技术研究与实现 [D]. 成都：电子科技大学, 2016.

[8] 杜永贵，陈鑫. 矩阵编码的遗传算法 [J]. 太原理工大学学报, 2012, 43 (2): 111-113, 118.

[9] 赵晓凡. 在线装箱问题相关近似算法研究 [D]. 北京：北京交通大学, 2016.

[10] 苏为华. 多指标综合评价理论与方法问题研究 [D]. 厦门：厦门大学, 2000.

[11] 葛继科，邱玉辉，吴春明，蒲国林. 遗传算法研究综述 [J]. 计算机应用研究, 2008, 35 (10): 2911-2916.

chinaXiv:201804.02162v1